Training NN

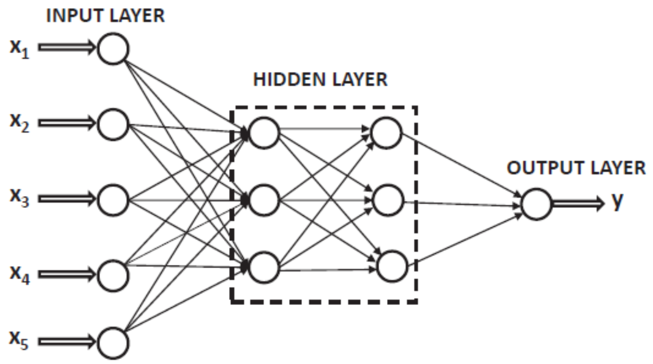
AW

Lecture Overview

1. Backpropagation I
2. Computational graphs
3. Backpropagation II

Data Set and Network Weights

Training data set: the set of training examples $S = \{(\vec{x}_i, y_i) | i \in 1, \dots, n\}$ where y is target variable



Network:

DAG of neurons divided into layers

- $W^{(i)}$ is weight matrix of neurons of layer i, so $\vec{w}_j^{(i)}$ is the column of input weights of neuron j of layer i with $w_{kj}^{(i)}$ weight of input of neuron k of layer $i-1$ to neuron j of layer i . All weights form 3-dimensional *tensor* $\{W^{(i)}\}_{i=1}^{k}$ denoted by $\boldsymbol{W}_{ijk}$ or just $\boldsymbol{W}$. Weights change during training

Note: *tensor* is just a fancy word for multidimensional array

Network, Data Set and training Goal

Training data set: the set of training examples $S = \{(\vec{x}_i, y_i) | i \in 1, \dots, n\}$ where y is target variable

Network:

- 3D weight tensor $\boldsymbol{W}$
- Activation functions of neurons of one layer are the same
- $\vec{\Phi}$ be he vector of activation functions, so that Φ_i is the activation function of layer i. Activation functions do not change during training.

Loss function: It is assumed to be cumulative, i.e.

$L_{cum}(S) = \sum_{i=1}^{n} L(p_{\overrightarrow{W}}(\vec{x}_i), y_i)$ where L is a loss function defined for one example x_i and $p_{\boldsymbol{W}}$ is a prediction function defined by $\boldsymbol{W}$ and $\vec{\Phi}$

Goal: To minimize loss function value L for the set of training examples S

Idea of Backpropagation

- Initialize input weights $\boldsymbol{W} = \left\{W^{(i)}\right\}_{i=1}^{k}$ for all neurons in all levels i from 1 to k.

```
Do
    For $i = 1, n$ take next sample $(\vec{x}_i, y_i) \in S$
        i.   In this step compute prediction $\hat{y}_i$ by feeding $x_i$ into the
             network using prediction function $p_{\boldsymbol{W}}$ defined by current
             weights $\overrightarrow{W}$.
        ii.  Compute the loss function $L_{\boldsymbol{W}}(\hat{y}_i, y_i)$
        iii. Compute the gradient $\nabla_{\boldsymbol{W}} L(\hat{y}_i, y_i)$ of the loss function $L$
             with respect to the different weights by using the chain rule
             of differential calculus.
        iv.  Compute new set of weights
             $$\boldsymbol{W}_{new} = \left\{W_{new}^{(i)} = W^{(i)} - \varepsilon \nabla_{\boldsymbol{W}} L(\hat{y}_i, y_i)\right\}_{i=1}^{k}$$
Until $\boldsymbol{W}_{new} = \boldsymbol{W}$
```

- ε is meta parameter chosen prior to training

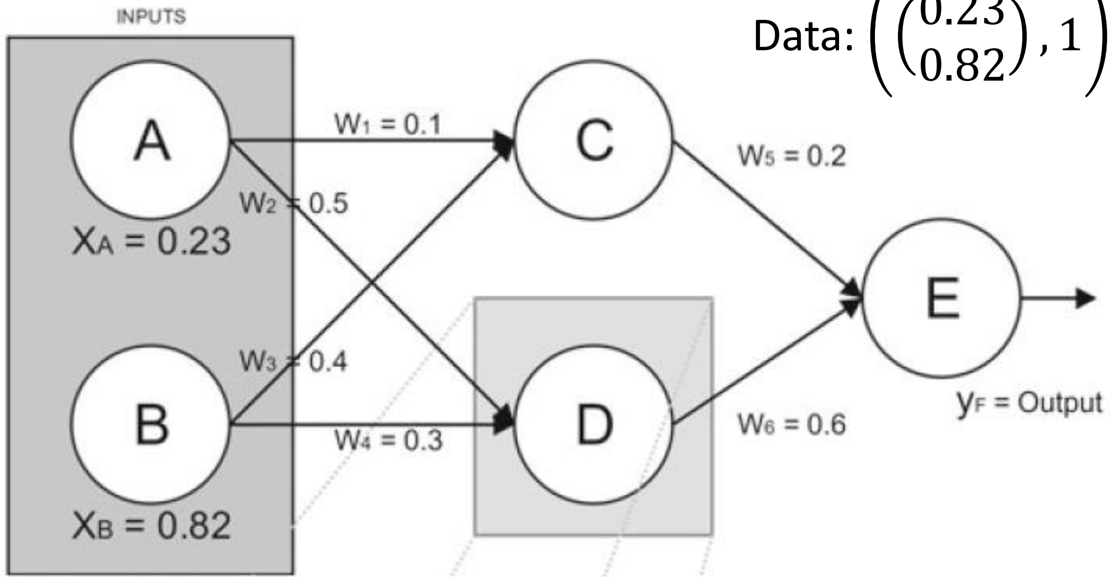
Naïve Computation of Gradient

- We can do steps i and ii in naïve way: compute output by propagating input through the network
- Compute loss function value
- Compute gradient by applying recursively chain formula for computing gradient as directed by the network
- Compute adjustments to weights given gradient

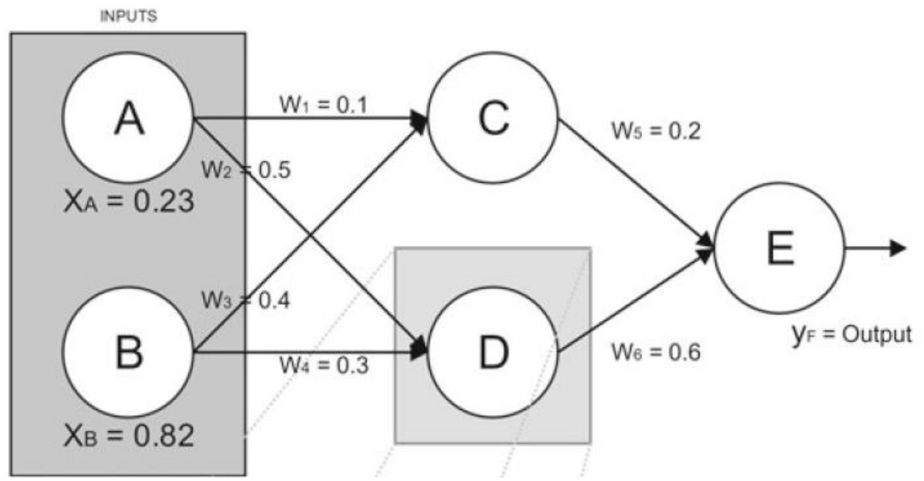
Example: we apply this method to NN below where all hidden nodes and output are sigmoids and learning rate is 0.7:

$y_n(z) = \frac{1}{1+e^{-z}}$;
$z = \vec{w}_n \cdot \vec{\iota}_n$ where $\vec{w}_n, \vec{\iota}_n$ are input weights and inputs to node n (no bias)
Loss is ½ SSE

Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$



Example: Forward Computation of Loss Function



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$$L = \frac{1}{2}\sum_{o\in Output}(y - \hat{y}_o)^2$$

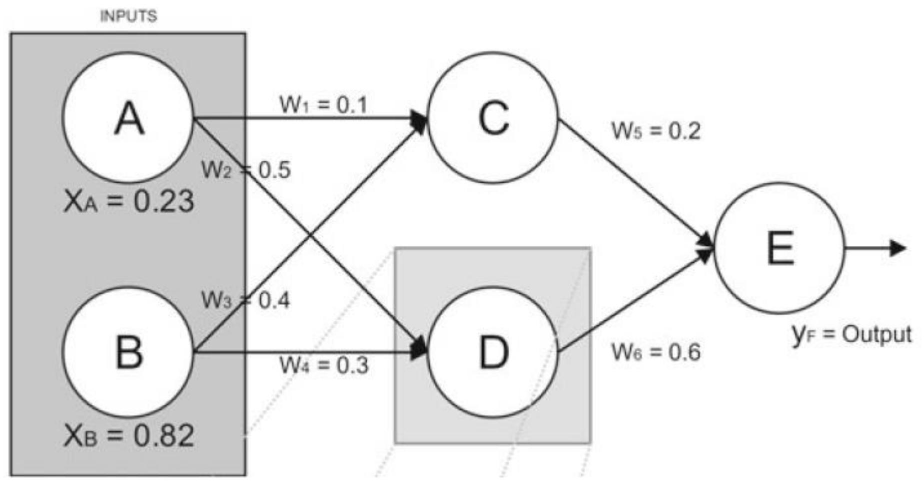
$$y_n(z) = \frac{1}{1+e^{-z}}$$

$$z = \vec{w}_n \cdot \vec{i}_n$$

So $\vec{w}_C = \begin{pmatrix}w_1\\w_3\end{pmatrix} = \begin{pmatrix}0.1\\0.4\end{pmatrix}$; $\vec{w}_D = \begin{pmatrix}w_2\\w_4\end{pmatrix} = \begin{pmatrix}0.5\\0.3\end{pmatrix}$; $\vec{w}_E = \begin{pmatrix}w_5\\w_6\end{pmatrix} = \begin{pmatrix}0.2\\0.6\end{pmatrix}$

First we need to compute output and loss.

Example: Forward Computation of Loss Function



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$$L = \frac{1}{2}\sum_{o\in Output}(y-\hat{y}_o)^2$$

$$y_n(z) = \frac{1}{1+e^{-z}}$$

$$z = \vec{w}_n \cdot \vec{i}_n$$

So $\vec{w}_C = \begin{pmatrix}w_1\\w_3\end{pmatrix} = \begin{pmatrix}0.1\\0.4\end{pmatrix}$; $\vec{w}_D = \begin{pmatrix}w_2\\w_4\end{pmatrix} = \begin{pmatrix}0.5\\0.3\end{pmatrix}$; $\vec{w}_E = \begin{pmatrix}w_5\\w_6\end{pmatrix} = \begin{pmatrix}0.2\\0.6\end{pmatrix}$

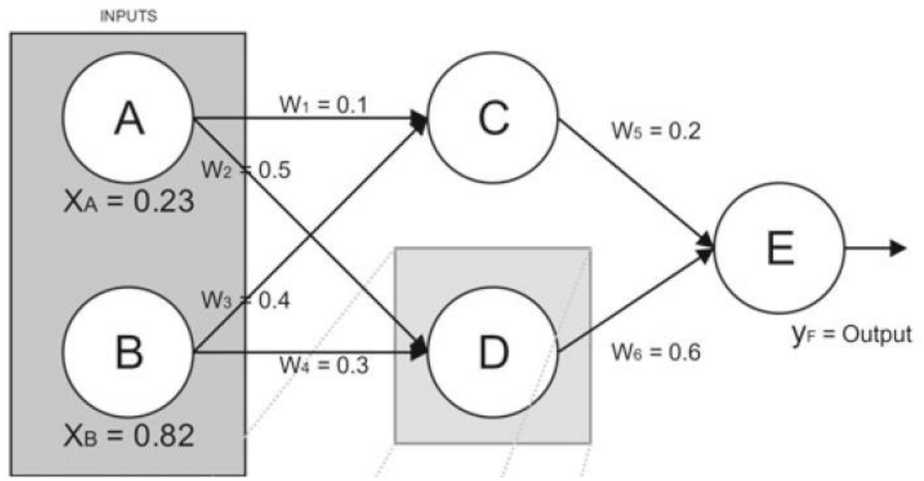
First we need to compute output and loss.

1. **For hidden layer activation input** is $\begin{pmatrix}z_C\\z_D\end{pmatrix} = W_h^T\vec{x}$. We have

$$W_h^T = (\vec{w}_C \;\; \vec{w}_D)^T = \begin{pmatrix}0.1 & 0.4\\0.5 & 0.3\end{pmatrix},$$

so $\begin{pmatrix}z_C\\z_D\end{pmatrix} = W_h^T\vec{x} = \begin{pmatrix}0.1 & 0.4\\0.5 & 0.3\end{pmatrix}\begin{pmatrix}0.23\\0.82\end{pmatrix} = \begin{pmatrix}0.351\\0.361\end{pmatrix}$

Example: Forward Computation of Loss Function



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$L = \frac{1}{2}\sum_{o\in Output}(y - \hat{y}_o)^2$

$y_n(z) = \frac{1}{1+e^{-z}}$

$z = \vec{w}_n \cdot \vec{i}_n$

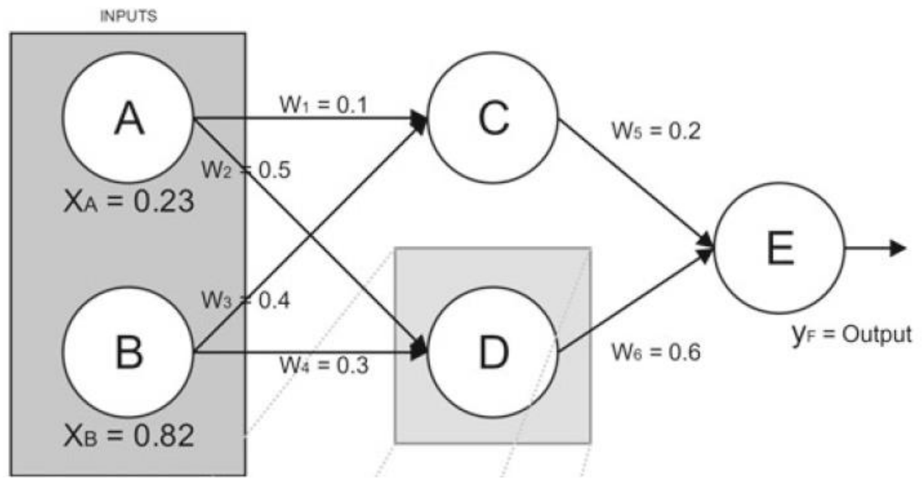
So $\vec{w}_C = \begin{pmatrix}w_1\\w_3\end{pmatrix} = \begin{pmatrix}0.1\\0.4\end{pmatrix}; \vec{w}_D = \begin{pmatrix}w_2\\w_4\end{pmatrix} = \begin{pmatrix}0.5\\0.3\end{pmatrix}; \vec{w}_E = \begin{pmatrix}w_5\\w_6\end{pmatrix} = \begin{pmatrix}0.2\\0.6\end{pmatrix}$

First we need to compute output and loss. For hidden layer activation input is $\begin{pmatrix}z_C\\z_D\end{pmatrix} = \begin{pmatrix}0.351\\0.361\end{pmatrix}$. Then

2. For hidden layer activation output is $\vec{y}_h = \begin{pmatrix}y_C\\y_D\end{pmatrix} = \frac{1}{1+e^{-\vec{z}}} = \begin{pmatrix}\frac{1}{1+e^{-z_C}}\\ \frac{1}{1+e^{-z_D}}\end{pmatrix}$, so

$$\vec{y}_h = \begin{pmatrix}\frac{1}{1+e^{-0.351}}\\ \frac{1}{1+e^{-0.361}}\end{pmatrix} = \begin{pmatrix}0.5868\\0.5892\end{pmatrix}$$

Example: Forward Computation of Loss Function



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$L = \frac{1}{2}\sum_{o\in Output}(y - \hat{y}_o)^2$

$y_n(z) = \frac{1}{1+e^{-z}}$

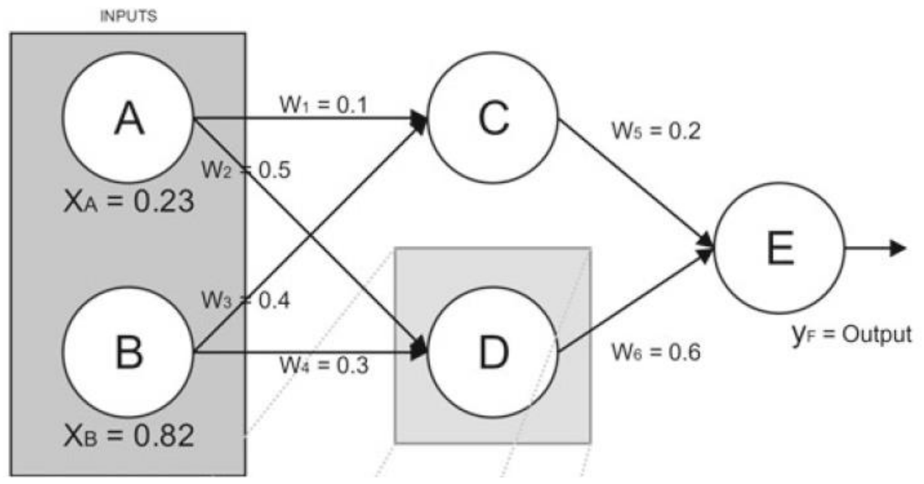
$z = \vec{w}_n \cdot \vec{\imath}_n$

So $\vec{w}_C = \begin{pmatrix}w_1\\w_3\end{pmatrix} = \begin{pmatrix}0.1\\0.4\end{pmatrix}$; $\vec{w}_D = \begin{pmatrix}w_2\\w_4\end{pmatrix} = \begin{pmatrix}0.5\\0.3\end{pmatrix}$; $\vec{w}_E = \begin{pmatrix}w_5\\w_6\end{pmatrix} = \begin{pmatrix}0.2\\0.6\end{pmatrix}$

3. For output layer activation input is $z_E = \vec{w}_E^T\vec{y}_h$ where we already computed

$\vec{y}_h = \begin{pmatrix}0.5868\\0.5892\end{pmatrix}$, so $z_E = \begin{pmatrix}0.2\\0.6\end{pmatrix}^T\begin{pmatrix}0.5868\\0.5892\end{pmatrix} = 0.4708$

Example: Forward Computation of Loss Function



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$$L = \frac{1}{2}\sum_{o \in Output}(y - \hat{y}_o)^2$$

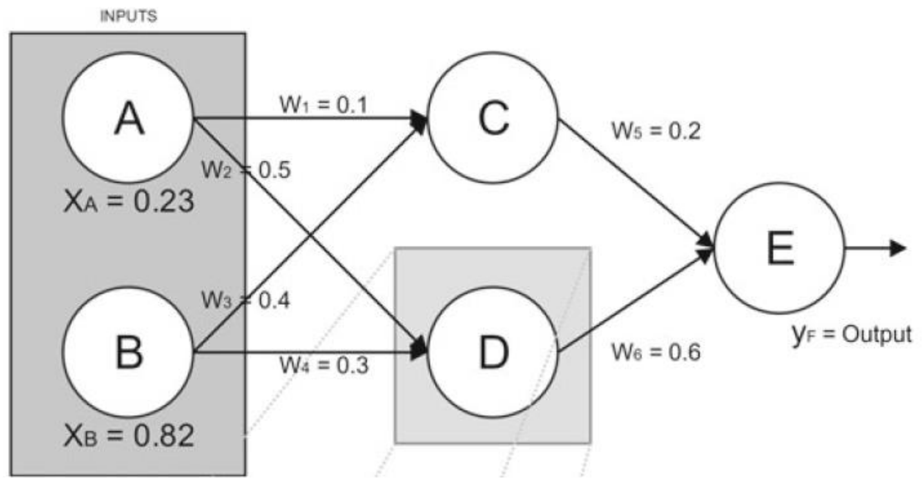
$$y_n(z) = \frac{1}{1+e^{-z}}$$

$$z = \vec{w}_n \cdot \vec{i}_n$$

So $\vec{w}_C = \begin{pmatrix}w_1\\w_3\end{pmatrix} = \begin{pmatrix}0.1\\0.4\end{pmatrix}; \vec{w}_D = \begin{pmatrix}w_2\\w_4\end{pmatrix} = \begin{pmatrix}0.5\\0.3\end{pmatrix}; \vec{w}_E = \begin{pmatrix}w_5\\w_6\end{pmatrix} = \begin{pmatrix}0.2\\0.6\end{pmatrix}$

4. For output layer activation output is $y_E = \frac{1}{1+e^{-z_E}}$ where we just computed $z_E = 0.4708$, so $y_E = \frac{1}{1+e^{-0.4708}} = 0.6155$

Example: Forward Computation of Loss Function



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$$L = \frac{1}{2}\sum_{o \in Output}(y - \hat{y}_o)^2$$

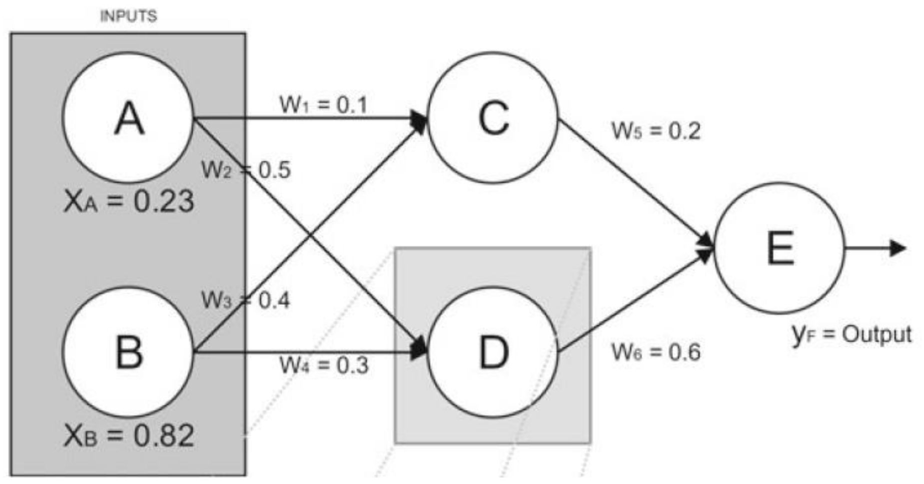
$$y_n(z) = \frac{1}{1+e^{-z}}$$

$$z = \vec{w}_n \cdot \vec{i}_n$$

So $\vec{w}_C = \begin{pmatrix}w_1\\w_3\end{pmatrix} = \begin{pmatrix}0.1\\0.4\end{pmatrix}$; $\vec{w}_D = \begin{pmatrix}w_2\\w_4\end{pmatrix} = \begin{pmatrix}0.5\\0.3\end{pmatrix}$; $\vec{w}_E = \begin{pmatrix}w_5\\w_6\end{pmatrix} = \begin{pmatrix}0.2\\0.6\end{pmatrix}$

5. The loss is 1/2SSE, so on 1 vector it is just $L = \frac{1}{2}(y - y_E)^2$ where we computed $y_E = 0.6155$, so $\frac{1}{2}(1 - 0.6155)^2 = 0.0739$

Summary: Forward Computation of Loss



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$$L = \frac{1}{2}\sum_{o \in Output}(y - \hat{y}_o)^2$$

$$y_n(z) = \frac{1}{1+e^{-z}}$$

$$z = \vec{w}_n \cdot \vec{i}_n$$

So $\vec{w}_C = \begin{pmatrix}w_1\\w_3\end{pmatrix} = \begin{pmatrix}0.1\\0.4\end{pmatrix}; \vec{w}_D = \begin{pmatrix}w_2\\w_4\end{pmatrix} = \begin{pmatrix}0.5\\0.3\end{pmatrix}; \vec{w}_E = \begin{pmatrix}w_5\\w_6\end{pmatrix} = \begin{pmatrix}0.2\\0.6\end{pmatrix}$

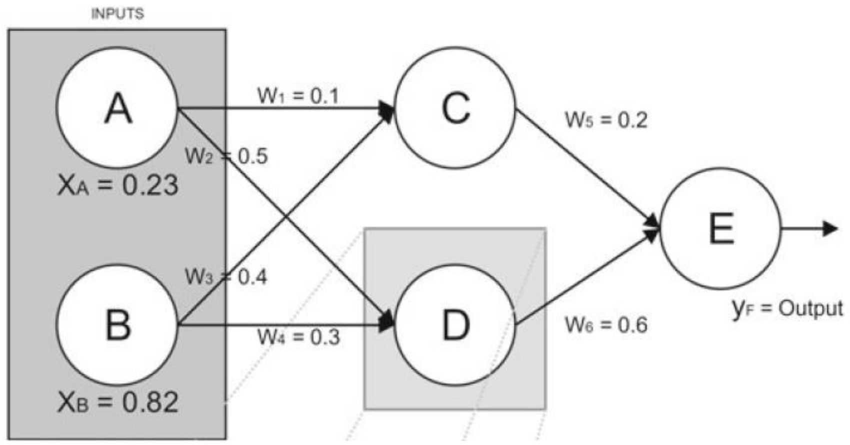
Activation input, and output of each layer and loss on input:

$$\vec{z}_h = \begin{pmatrix}0.351\\0.361\end{pmatrix},\ \vec{y}_h = \begin{pmatrix}y_C\\y_D\end{pmatrix} = \begin{pmatrix}0.5868\\0.5892\end{pmatrix}$$

$$z_E = 0.4708,\ \hat{y}_E = 0.6155$$

$$L = 0.0739$$

Example: Backward Computation of Gradient



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$L = \frac{1}{2}\sum_{o \in Output}(\hat{y}_o - y)^2$

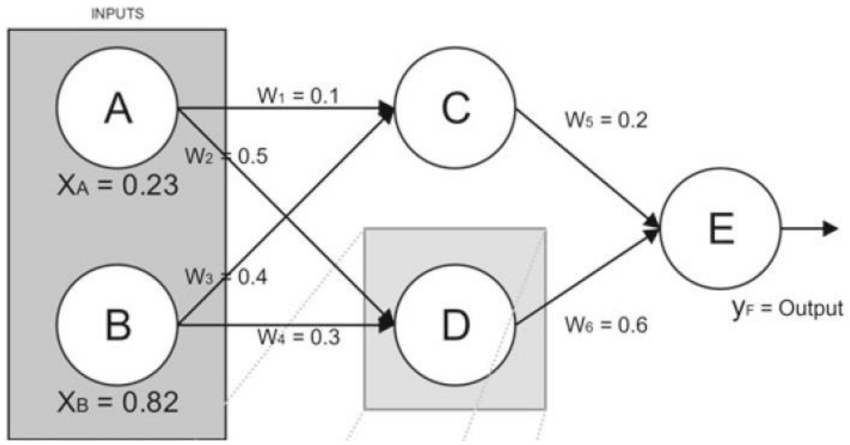
$y_n(z) = \frac{1}{1+e^{-z}}$

$z = \vec{w}_n \cdot \vec{i}_n$

$$\nabla_{\vec{w}} L = \begin{pmatrix} \frac{\partial L}{\partial w_1} & \frac{\partial L}{\partial w_2} & \frac{\partial L}{\partial w_3} & \frac{\partial L}{\partial w_4} & \frac{\partial L}{\partial w_5} & \frac{\partial L}{\partial w_6} \end{pmatrix}^T$$

Apply chain rule to the to $L = \frac{1}{2}(y - y_E)^2$ so first derivative is over y_E

Example: Backward Computation of Gradient



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$$L = \frac{1}{2}\sum_{o\in Output}(\hat{y}_o - y)^2$$

$$y_n(z) = \frac{1}{1+e^{-z}}$$

$$z = \vec{w}_n \cdot \vec{i}_n$$

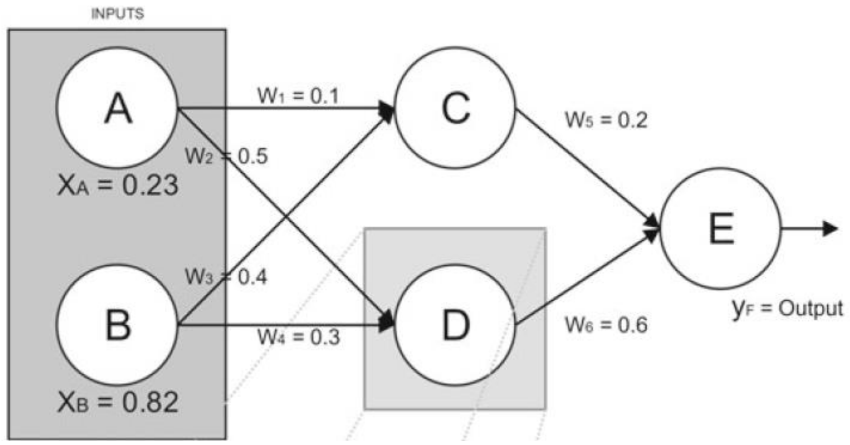
$$\nabla_{\vec{w}} L = \begin{pmatrix}\frac{\partial L}{\partial w_1} & \frac{\partial L}{\partial w_2} & \frac{\partial L}{\partial w_3} & \frac{\partial L}{\partial w_4} & \frac{\partial L}{\partial w_5} & \frac{\partial L}{\partial w_6}\end{pmatrix}^T$$

Apply chain rule to the to $L = \frac{1}{2}(y - y_E)^2$. First derivative is over y_E, so by chain rule

$$= \frac{\partial \mathrm{L}}{\partial y_E} \cdot \begin{pmatrix}\frac{\partial y_E}{\partial w_1} & \frac{\partial y_E}{\partial w_2} & \frac{\partial y_E}{\partial w_3} & \frac{\partial y_E}{\partial w_4} & \frac{\partial y_E}{\partial w_5} & \frac{\partial y_E}{\partial w_6}\end{pmatrix}^T$$

Next derivative is over z_E

Example: Backward Computation of Gradient



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$$L = \frac{1}{2}\sum_{o\in Output}(\hat{y}_o - y)^2$$

$$y_n(z) = \frac{1}{1+e^{-z}}$$

$$z = \vec{w}_n \cdot \vec{i}_n$$

$$\nabla_{\vec{w}} L = \begin{pmatrix}\frac{\partial L}{\partial w_1} & \frac{\partial L}{\partial w_2} & \frac{\partial L}{\partial w_3} & \frac{\partial L}{\partial w_4} & \frac{\partial L}{\partial w_5} & \frac{\partial L}{\partial w_6}\end{pmatrix}^T$$

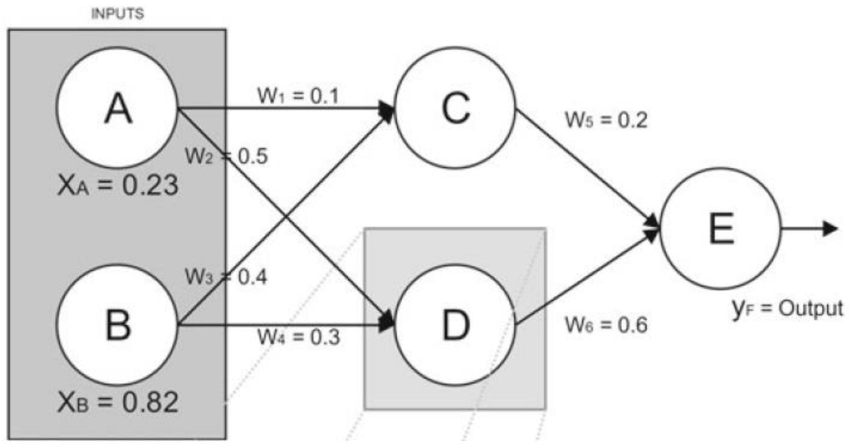
$$= \frac{\partial \mathrm{L}}{\partial y_E} \cdot \begin{pmatrix}\frac{\partial y_E}{\partial w_1} & \frac{\partial y_E}{\partial w_2} & \frac{\partial y_E}{\partial w_3} & \frac{\partial y_E}{\partial w_4} & \frac{\partial y_E}{\partial w_5} & \frac{\partial y_E}{\partial w_6}\end{pmatrix}^T$$

Next derivative is over z_E

$$= \frac{\partial \mathrm{L}}{\partial y_E} \cdot \frac{\partial y_E}{\partial z_E} \cdot \begin{pmatrix}\frac{\partial z_E}{\partial w_1} & \frac{\partial z_E}{\partial w_2} & \frac{\partial z_E}{\partial w_3} & \frac{\partial z_E}{\partial w_4} & \frac{\partial z_E}{\partial w_5} & \frac{\partial z_E}{\partial w_6}\end{pmatrix}^T$$

$$= \frac{\partial \mathrm{L}}{\partial y_E} \cdot \frac{\partial y_E}{\partial z_E} \nabla_{\vec{w}} z_E$$

Example: Backward Computation of Gradient



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$L = \frac{1}{2}\sum_{o \in Output}(\hat{y}_o - y)^2$

$y_n(z) = \frac{1}{1+e^{-z}}$

$z = \vec{w}_n \cdot \vec{i}_n$

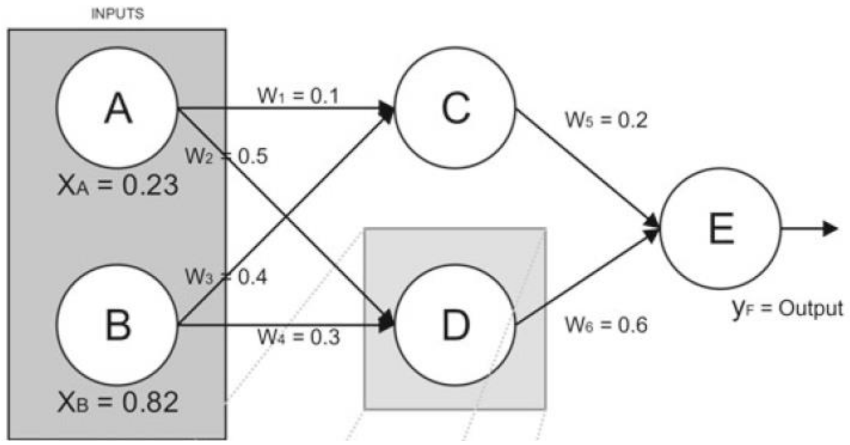
$$\nabla_{\vec{w}} L = \left(\frac{\partial L}{\partial w_1} \quad \frac{\partial L}{\partial w_2} \quad \frac{\partial L}{\partial w_3} \quad \frac{\partial L}{\partial w_4} \quad \frac{\partial L}{\partial w_5} \quad \frac{\partial L}{\partial w_6}\right)^T$$

$$= \frac{\partial \mathrm{L}}{\partial y_E} \cdot \frac{\partial \mathrm{y_E}}{\partial z_E} \nabla_{\vec{w}} z_E$$

where $z_E = \vec{w}_E^T \vec{y}_h = \begin{pmatrix}w_5\\w_6\end{pmatrix}^T \begin{pmatrix}y_c\\y_d\end{pmatrix} = w_5 y_C + w_6 y_D$, so $z_E = g_1(\vec{w}) + g_2(\vec{w})$

where $g_1(\vec{w}) = w_5 y_C$ and $g_2(\vec{w})$= $w_6 y_D$ so by chain rule $\nabla_{\vec{w}} z_E = (\mathbb{J}_{\vec{w}} z_E)^T . \nabla_{\vec{g}} z_E$

Example: Backward Computation of Gradient



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$L = \frac{1}{2}\sum_{o\in Output}(\hat{y}_o - y)^2$

$y_n(z) = \frac{1}{1+e^{-z}}$

$z = \vec{w}_n \cdot \vec{i}_n$

$$\nabla_{\vec{w}}L = \frac{\partial \mathrm{L}}{\partial y_E} \cdot \frac{\partial \mathrm{y_E}}{\partial z_E}\nabla_{\vec{w}}z_E = \frac{\partial \mathrm{L}}{\partial y_E} \cdot \frac{\partial \mathrm{y_E}}{\partial z_E} \cdot (\mathbb{J}_{\vec{w}}z_E)^T . \nabla_{\vec{g}}z_E$$

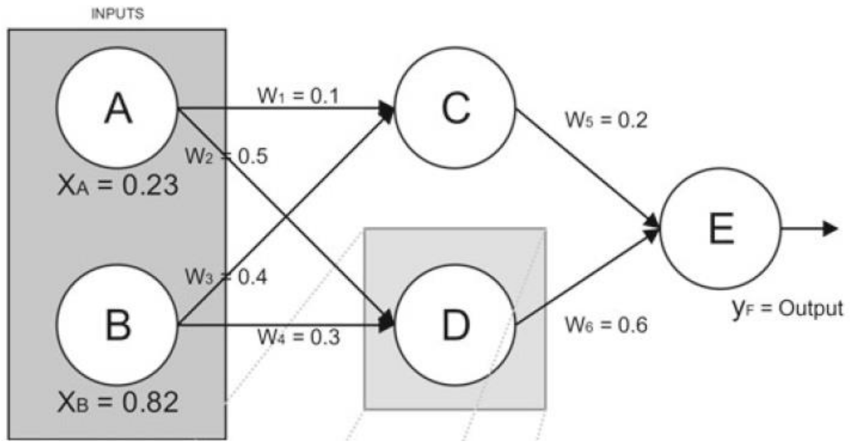
where $z_E = \vec{w}_E^T\vec{y}_h = g_1(\vec{w}) + g_2(\vec{w}) = w_5y_C + w_6y_D$, where $g_1(\vec{w}) = w_5y_C$ and $g_2(\vec{w})$= w_6y_D

Need to find $\mathbb{J}_{\vec{w}}z_E$ and $\nabla_{\vec{g}}z_E$. The latter is

$$\nabla_{\vec{g}}(g_1 + g_2) = \begin{pmatrix}\frac{\partial}{\partial g_1}(g_1+g_2)\\ \frac{\partial}{\partial g_2}(g_1+g_2)\end{pmatrix} = \begin{pmatrix}1\\1\end{pmatrix}$$

Substitute it into $\nabla_{\vec{w}}L$

Example: Backward Computation of Gradient



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$L = \frac{1}{2}\sum_{o\in Output}(\hat{y}_o - y)^2$

$y_n(z) = \frac{1}{1+e^{-z}}$

$z = \vec{w}_n \cdot \vec{i}_n$

$$\nabla_{\vec{w}} L = \frac{\partial \mathrm{L}}{\partial y_E} \cdot \frac{\partial \mathrm{y_E}}{\partial z_E} \nabla_{\vec{w}} z_E = \frac{\partial \mathrm{L}}{\partial y_E} \cdot \frac{\partial \mathrm{y_E}}{\partial z_E} \cdot (\mathbb{J}_{\vec{w}} z_E)^T \begin{pmatrix}1\\1\end{pmatrix}$$

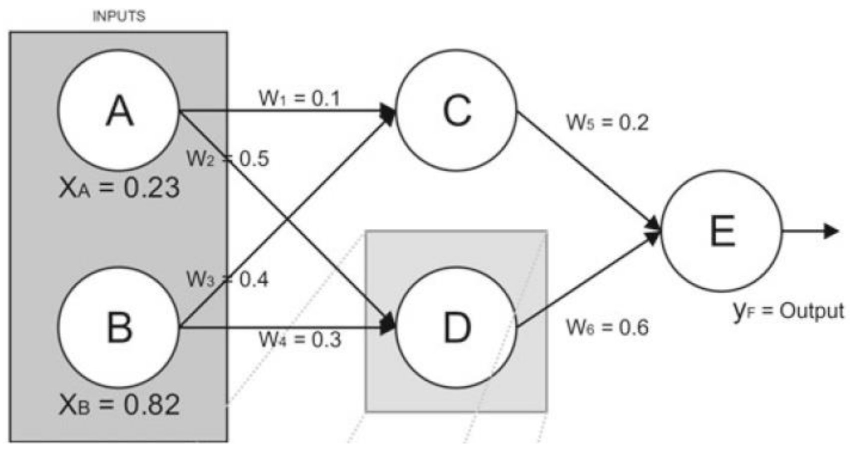
where $z_E = \vec{w}_E^T \vec{y}_h = g_1(\vec{w}) + g_2(\vec{w}) = w_5 y_C + w_6 y_D$, where $g_1(\vec{w}) = w_5 y_C$ and $g_2(\vec{w})$= $w_6 y_D$

Need to find

$$\mathbb{J}_{\vec{w}} z_E = \mathbb{J}_{\vec{w}}(w_5 y_C + w_6 y_D)$$

$$= \begin{pmatrix}\nabla_{\vec{w}}(w_5 y_c)\\ \nabla_{\vec{w}}(w_6 y_D)\end{pmatrix}$$

Example: Backward Computation of Gradient



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$L = \frac{1}{2}\sum_{o \in Output}(\hat{y}_o - y)^2$

$y_n(z) = \frac{1}{1+e^{-z}}$

$z = \vec{w}_n \cdot \vec{i}_n$

Need to find $\mathbb{J}_{\vec{w}} z_E = \begin{pmatrix}\nabla_{\vec{w}}(w_5 y_c)\\ \nabla_{\vec{w}}(w_6 y_D)\end{pmatrix}$

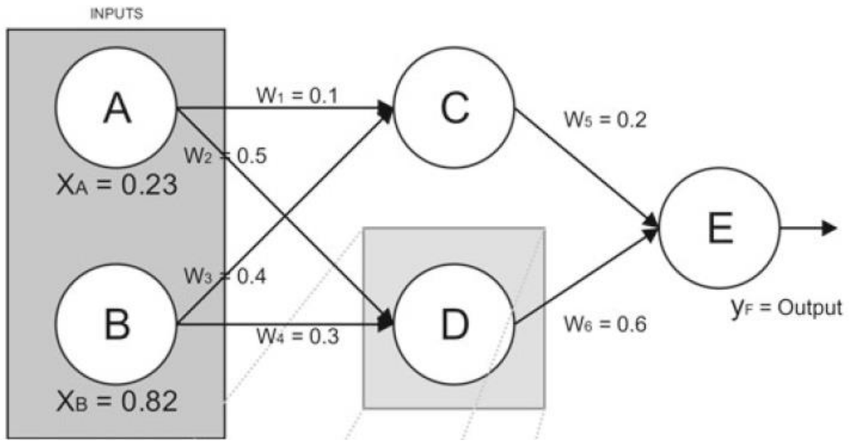
$$= \begin{pmatrix} \frac{\partial}{\partial w_1}(w_5 y_C) & \frac{\partial}{\partial w_2}(w_5 y_C) & \frac{\partial}{\partial w_3}(w_5 y_C) & \frac{\partial}{\partial w_4}(w_5 y_C) & \frac{\partial}{\partial w_5}(w_5 y_C) & \frac{\partial}{\partial w_6}(w_5 y_C) \\ \frac{\partial}{\partial w_1}(w_6 y_D) & \frac{\partial}{\partial w_2}(w_6 y_D) & \frac{\partial}{\partial w_3}(w_6 y_D) & \frac{\partial}{\partial w_4}(w_6 y_D) & \frac{\partial}{\partial w_5}(w_6 y_D) & \frac{\partial}{\partial w_6}(w_6 y_D) \end{pmatrix}$$

Note that y_c depend only on w_1 and w_3 while y_D depends only on w_4 and w_2, so

$$\frac{\partial}{\partial w_2}(w_5 y_C) = \frac{\partial}{\partial w_4}(w_5 y_C) = \frac{\partial}{\partial w_6}(w_5 y_C) = 0$$

$$\frac{\partial}{\partial w_1}(w_6 y_D) = \frac{\partial}{\partial w_3}(w_6 y_D) = \frac{\partial}{\partial w_5}(w_6 y_D) = 0$$

Example: Backward Computation of Gradient



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$L = \frac{1}{2}\sum_{o \in Output}(\hat{y}_o - y)^2$

$y_n(z) = \frac{1}{1+e^{-z}}$

$z = \vec{w}_n \cdot \vec{i}_n$

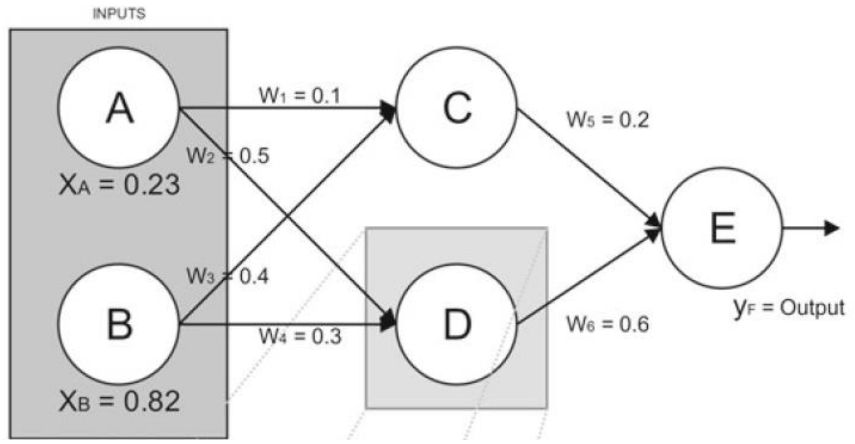
Need to find

$$\mathbb{J}_{\vec{w}} z_E = \begin{pmatrix} \frac{\partial}{\partial w_1}(w_5 y_C) & 0 & \frac{\partial}{\partial w_3}(w_5 y_C) & 0 & y_C & 0 \\ 0 & \frac{\partial}{\partial w_2}(w_6 y_D) & 0 & \frac{\partial}{\partial w_4}(w_6 y_D) & 0 & y_D \end{pmatrix}$$

Since $[cf(x)]'_x = c[f(x)]'_x$ we have

$$\mathbb{J}_{\vec{w}} z_E = \begin{pmatrix} w_5 \frac{\partial y_C}{\partial w_1} & 0 & w_5 \frac{\partial y_C}{\partial w_3} & 0 & y_C & 0 \\ 0 & w_6 \frac{\partial y_D}{\partial w_2} & 0 & w_6 \frac{\partial y_D}{\partial w_4} & 0 & y_D \end{pmatrix}$$

Example: Backward Computation of Gradient



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$$L = \frac{1}{2}\sum_{o\in Output}(\hat{y}_o - y)^2$$

$$y_n(z) = \frac{1}{1+e^{-z}}$$

$$z = \vec{w}_n \cdot \vec{i}_n$$

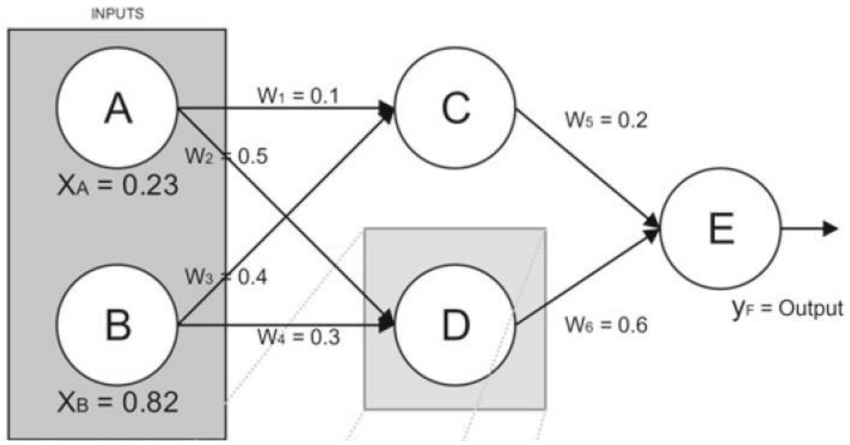
Need to find $\mathbb{J}_{\vec{w}} z_E = \begin{pmatrix} w_5 \frac{\partial y_C}{\partial w_1} & 0 & w_5 \frac{\partial y_C}{\partial w_3} & 0 & y_C & 0 \\ 0 & w_6 \frac{\partial y_D}{\partial w_2} & 0 & w_6 \frac{\partial y_D}{\partial w_4} & 0 & y_D \end{pmatrix}$

By chain rule $\frac{\partial y_C}{\partial w_1} = \frac{\partial y_C}{\partial z_C} \cdot \frac{\partial z_C}{\partial w_1}, \frac{\partial y_C}{\partial w_3} = \frac{\partial y_C}{\partial z_C} \cdot \frac{\partial z_C}{\partial w_3}$ and

$\frac{\partial y_D}{\partial w_2} = \frac{\partial y_D}{\partial z_D} \cdot \frac{\partial z_D}{\partial w_2}, \frac{\partial y_D}{\partial w_4} = \frac{\partial y_D}{\partial z_D} \cdot \frac{\partial z_D}{\partial w_4}$, so substituting

$$\mathbb{J}_{\vec{w}} z_E = \begin{pmatrix} w_5 \frac{\partial y_C}{\partial z_C} \cdot \frac{\partial z_C}{\partial w_1} & 0 & w_5 \frac{\partial y_C}{\partial z_C} \cdot \frac{\partial z_C}{\partial w_3} & 0 & y_C & 0 \\ 0 & w_6 \frac{\partial y_D}{\partial z_D} \cdot \frac{\partial z_D}{\partial w_2} & 0 & w_6 \frac{\partial y_D}{\partial z_D} \cdot \frac{\partial z_D}{\partial w_4} & 0 & y_D \end{pmatrix}$$

Example: Backward Computation of Gradient



Data: $\vec{x}_d = \left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$\vec{z}_h = \begin{pmatrix}0.351\\0.361\end{pmatrix}$, $\vec{y}_h = \begin{pmatrix}y_C\\y_D\end{pmatrix} = \begin{pmatrix}0.5868\\0.5892\end{pmatrix}$

$z_E = 0.4708,\ \hat{y}_E = 0.6155$

Putting everything back together again we have

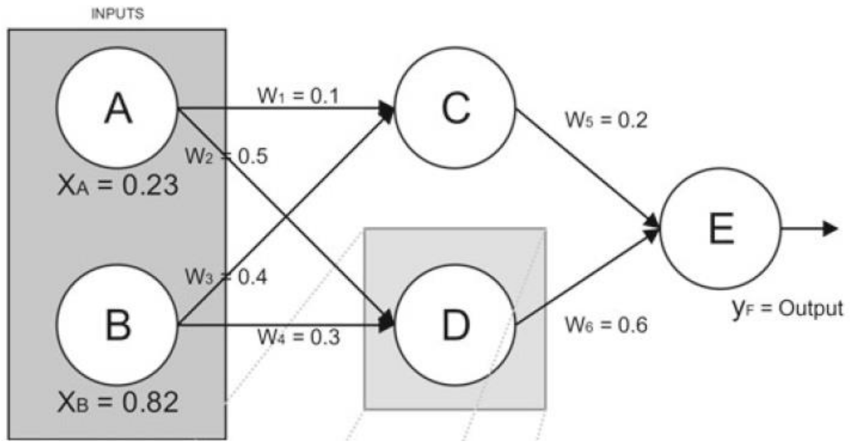
$$\nabla_{\vec{w}} L = \frac{\partial \mathrm{L}}{\partial y_E} \cdot \frac{\partial \mathrm{y_E}}{\partial z_E} \cdot \begin{pmatrix} w_5 \frac{\partial y_C}{\partial z_C} \cdot \frac{\partial z_C}{\partial w_1} & 0 & w_5 \frac{\partial y_C}{\partial z_C} \cdot \frac{\partial z_C}{\partial w_3} & 0 & y_C & 0 \\ 0 & w_6 \frac{\partial y_D}{\partial z_D} \cdot \frac{\partial z_D}{\partial w_2} & 0 & w_6 \frac{\partial y_D}{\partial z_D} \cdot \frac{\partial z_D}{\partial w_4} & 0 & y_D \end{pmatrix}^T \begin{pmatrix}1\\1\end{pmatrix}$$

We need $\nabla_{\vec{w}} L|_{x_d}$ where every derivative can be computed directly:

$$\frac{\partial \mathrm{L}}{\partial y_E} = \left(\frac{1}{2}(1-y_E)^2\right)'_{y_E}\Bigg|_{y_E(\vec{x}_d)} = -1 + y_E|_{y_E(\vec{x}_d)} = -1 + 0.6155 = -0.3845$$

$$\frac{\partial \mathrm{y_E}}{\partial z_E}\Bigg|_{z_E(\vec{x}_d)} = \frac{e^{-z_E}}{(1+e^{-z_E})^2}\Bigg|_{z_E(\vec{x}_d)} = (1-y_E)y_E = 0.6155 \cdot 0.3845 = 0.2365$$

Example: Backward Computation of Gradient



Data: $\vec{x}_d = \left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$\vec{z}_h = \begin{pmatrix}0.351\\0.361\end{pmatrix}$, $\vec{y}_h = \begin{pmatrix}y_C\\y_D\end{pmatrix} = \begin{pmatrix}0.5868\\0.5892\end{pmatrix}$

$z_E = 0.4708,\ \hat{y}_E = 0.6155$

Putting everything back together again we have

$$\nabla_{\vec{w}} L = -0.3845 \times 0.2365 \times$$

$$\times \begin{pmatrix} 0.2\frac{\partial y_C}{\partial z_C}\cdot\frac{\partial z_C}{\partial w_1} & 0 & 0.2\frac{\partial y_C}{\partial z_C}\cdot\frac{\partial z_C}{\partial w_3} & 0 & 0.5868 & 0 \\ 0 & 0.6\frac{\partial y_D}{\partial z_D}\cdot\frac{\partial z_D}{\partial w_2} & 0 & 0.6\frac{\partial y_D}{\partial z_D}\cdot\frac{\partial z_D}{\partial w_4} & 0 & 0.5892 \end{pmatrix}^T \times \begin{pmatrix}1\\1\end{pmatrix}$$

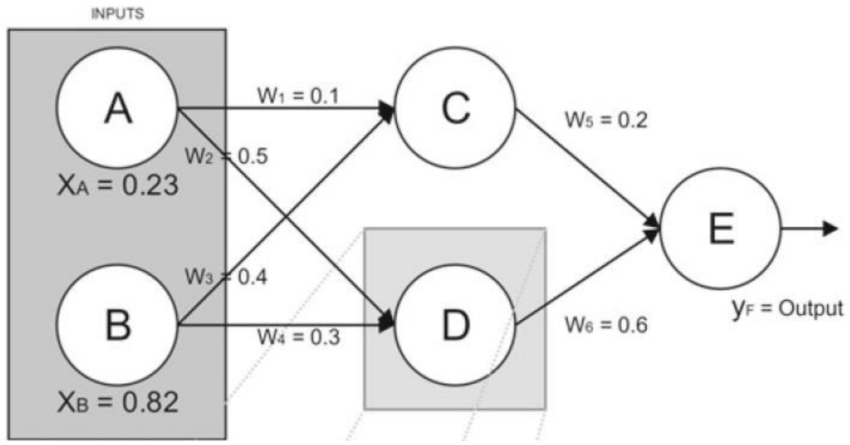
where $\left.\frac{\partial y_C}{\partial z_C}\right|_{z_c(\vec{x}_d)} \cdot \left.\frac{\partial z_C}{\partial w_1}\right|_{\vec{x}_d} = (1-y_c)y_c \cdot x_A = (1-0.351)\cdot 0.351 \cdot 0.23 = 0.0524$

$$\left.\frac{\partial y_C}{\partial z_C}\right|_{z_c(\vec{x}_d)} \cdot \left.\frac{\partial z_C}{\partial w_3}\right|_{\vec{x}_d} = (1-y_c)y_c \cdot x_B = (1-0.351)\cdot 0.351 \cdot 0.82 = 0.1868$$

$$\left.\frac{\partial y_D}{\partial z_D}\right|_{z_c(\vec{x}_d)} \cdot \left.\frac{\partial z_D}{\partial w_2}\right|_{\vec{x}_d} = (1-y_D)y_D \cdot x_A = (1-0.361)\cdot 0.361 \cdot 0.23 = 0.0531$$

$$\left.\frac{\partial y_D}{\partial z_D}\right|_{z_c(\vec{x}_d)} \cdot \left.\frac{\partial z_D}{\partial w_2}\right|_{\vec{x}_d} = (1-y_D)y_D \cdot x_A = (1-0.361)\cdot 0.361 \cdot 0.82 = 0.1892$$

Example: Backward Computation of Gradient



Data: $\vec{x}_d = \left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$\vec{z}_h = \begin{pmatrix}0.351\\0.361\end{pmatrix}$, $\vec{y}_h = \begin{pmatrix}y_C\\y_D\end{pmatrix} = \begin{pmatrix}0.5868\\0.5892\end{pmatrix}$

$z_E = 0.4708,\ \hat{y}_E = 0.6155$

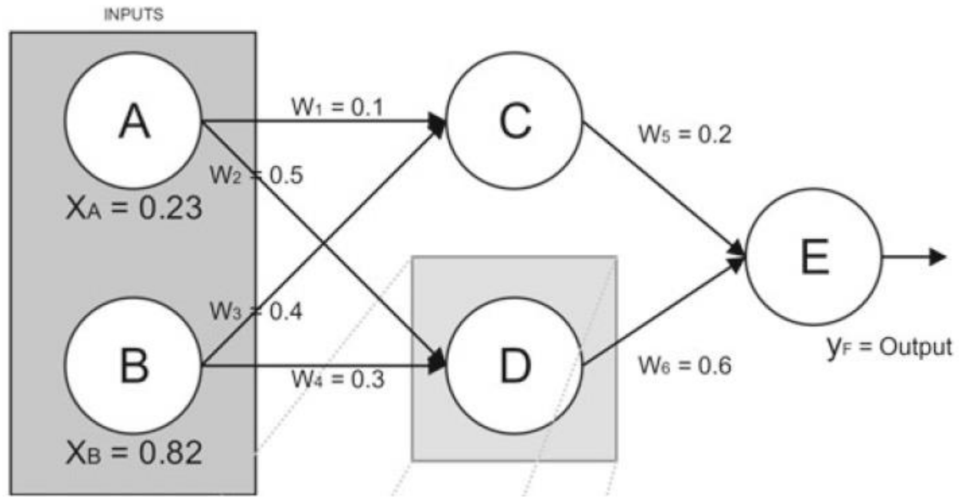
Putting everything back together again we have

$$\nabla_{\vec{w}} L = -0.3845 \times 0.2365 \times$$

$$\times \begin{pmatrix}0.01048 & 0 & 0.03736 & 0 & 0.5868 & 0\\ 0 & 0.03186 & 0 & 0.1155 & 0 & 0.5892\end{pmatrix}^T \times \begin{pmatrix}1\\1\end{pmatrix}$$

$$= -0.0909 \times \begin{pmatrix}0.01048\\0.03186\\0.03736\\0.11551\\0.5868\\0.5892\end{pmatrix} = \begin{pmatrix}-0.000952632\\-0.00289062\\-0.00339602\\-0.010499\\-0.0533401\\-0.0535583\end{pmatrix}$$

Example: Forward Computation of Loss Function



Data: $\left(\begin{pmatrix}0.23\\0.82\end{pmatrix}, 1\right)$

$$L = \frac{1}{2}\sum_{o\in Output}(y - \hat{y}_o)^2$$

$$y_n(z) = \frac{1}{1+e^{-z}}$$

$$z = \vec{w}_n \cdot \vec{i}_n$$

So assuming $\varepsilon = 0.7$ and using

$$\vec{w}' = \vec{w} - 0.7 \cdot \nabla_{\vec{w}}\left(\frac{1}{2}(y - y_E(\vec{x},\vec{w}))^2\right|_{(1,0.23,0.82,0.1,0.5,0.4,.0.3,0.2,0.6)^T}$$

we obtain after one iteration on a given data new set of weights

$$\vec{w}' = \begin{pmatrix}0.1\\0.5\\0.4\\0.3\\0.2\\0.6\end{pmatrix} - 0.7\begin{pmatrix}-0.000952632\\-0.00289062\\-0.00339602\\-0.010499\\-0.0533401\\-0.0535583\end{pmatrix} \approx \begin{pmatrix}0.1007\\0.5020\\0.4024\\0.3073\\0.2373\\0.6375\end{pmatrix}$$

Lecture Overview

1. Backpropagation I

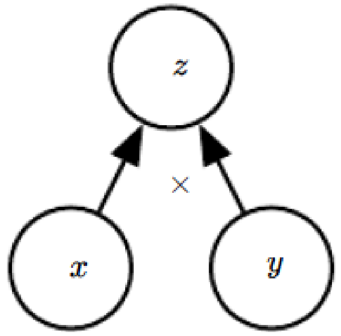
2. **Computational graphs**

3. Backpropagation II

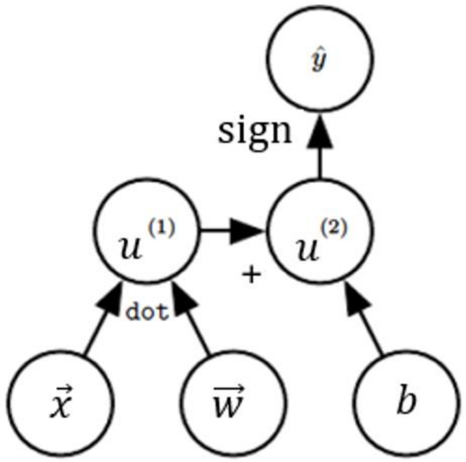
Computational Graph Language

- Directed labeled graphs
  - Nodes labeled by a pair (x, op) where
  - x is a variable, op is an operation
- Variables that may have types, e.g. scalar, vector, matrix, k-dimensional tensor
- Operations are (symmetric) functions with arguments that are given by incoming edges
- Informally, if a variable y is computed by applying an operation to variables among which is a variable x, then we draw a directed edge from x to y.

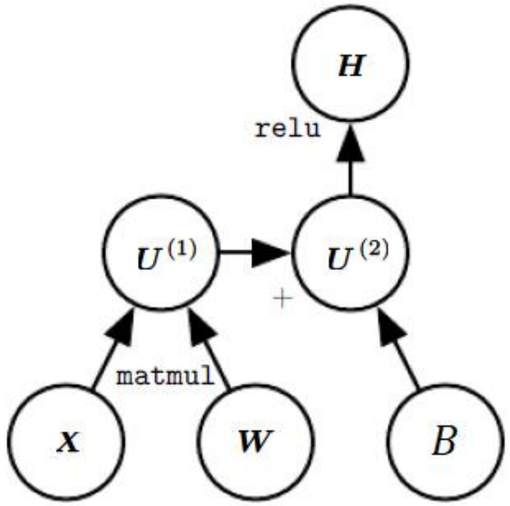
Examples of Computational Graphs



Graph for $z = x \cdot y$, all variables are scalars



Graph for $\hat{y} = \text{sign}(\vec{w} \cdot \vec{x} + b)$



Graph that computes $H = \max\{0, XW + B\}$ where $X, W, B, U^{(1)}, U^{(2)}, H$ and 0 are all matrices

NN as Computational Graphs

- NN clearly can be computed to a *computational graph*, in which each neuron can be represented as a unit of computation
- Operations are
  - all operations on numbers (scalar operations
  - dot product, vector addition, multiplication of a vector by a scalar
  - Matrix multiplication, transpose, determinant, Hadamard multiplication and Kronecker multiplication of matrices -respectively

$$A * B = \begin{pmatrix} a_{11}b_{11} & \dots & a_{1n}b_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & \dots & a_{mn}b_{mn} \end{pmatrix}, \mathrm{A} \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{pmatrix}$$

  - sigmoid, tanh, hard tanh, ReLU, Softmax

Lecture Overview

1. Gradient-based optimization
2. Backpropagation I
3. Computational graphs
4. **Backpropagation II**

Network to Train, Data Set and training Goal

Still the same input:

Training data set: the set of training examples $S = \{(\vec{x}_i, y_i) | i \in 1, \dots, n\}$ where y is target variable

Network:

- $W^{(i)}$ is weight matrix of neurons of level i, so $\vec{w}_{ij}$ is the column of weights of neuron j of level i. All weights are 3-dimensional *tensor* $\{W^{(i)}\}_{i=1}^{k}$ denoted by $\boldsymbol{W}_{ijk}$ or just $\boldsymbol{W}$. Weight are considered variable in training
- $\vec{\Phi}$ be he vector of activation functions, so that Φ_i is the activation function of layer i. Activation functions do not change.

Loss function: It is assumed to be cumulative, i.e.

$L_{cum}(S) = \sum_{i=1}^{n} L(p_{\vec{W}}(\vec{x}_i), y_i)$ where L is a loss function define for one example and $p_{\boldsymbol{W}}$ is a prediction function defined by $\boldsymbol{W}$ and $\vec{\Phi}$

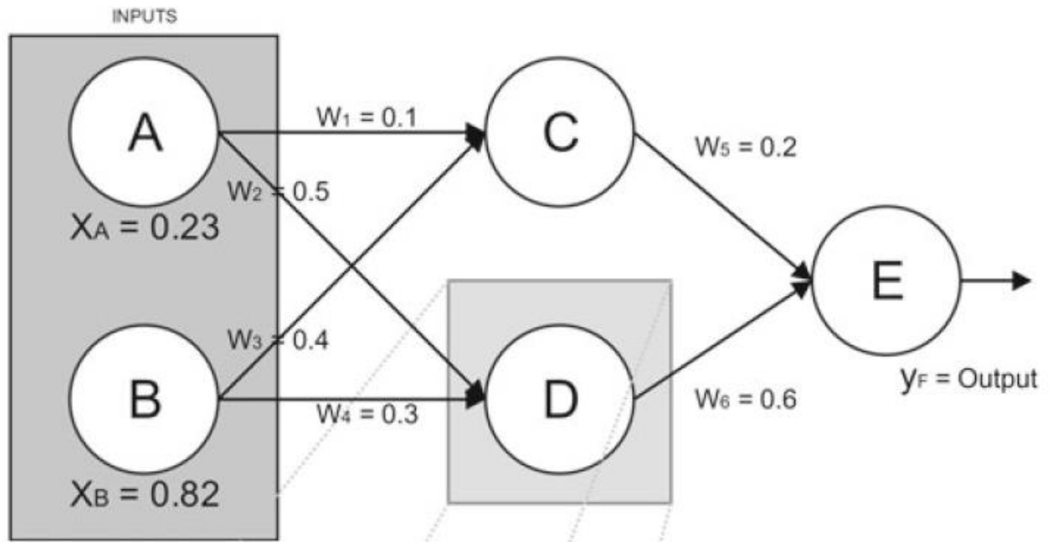
Goal:

- To minimize loss function value L for the set of training examples S

Computational Graphs – Labeling Agreements

- We need an algorithm to build a computational graph G corresponding to our NN that describes how to compute a single scalar $u^{(n)}$ (that is the loss on a training example).
- We need to construct a graph in which all nodes are labeled by variables $u^{(1)}, \dots, u^{(n)}$ in such a ways that first nodes $u^{(1)}, \dots, u^{(l_1)}$ are inputs, $u^{(l_1+1)}, \dots, u^{(l_2)}$ label first hidden layer that is fed by inputs, nodes labeled by $u^{(l_{i-1}+1)}, \dots, u^{(l_i)}$ belong to layer i and are fed by outputs of previous layer, so if there are $k-1$ layers in the computational graph then a node labled by u_n can take inputs from nodes labeled $u^{(l_{k-1}+1)}, \dots, u^{(l_k)}$.
- Each variable $u^{(i)}$ is associated with an operation $f^{(i)}$ labeling the same node and is computed by evaluating the function $u^{(i)} = f(\mathbb{A}^{(i)})$ where $\mathbb{A}^{(i)}$ is the set of variables of all nodes that are members of the set $Pa(i)$ of parents of the node labeled with variable $u^{(i)}$.

Conversion of NN to Computational Graph

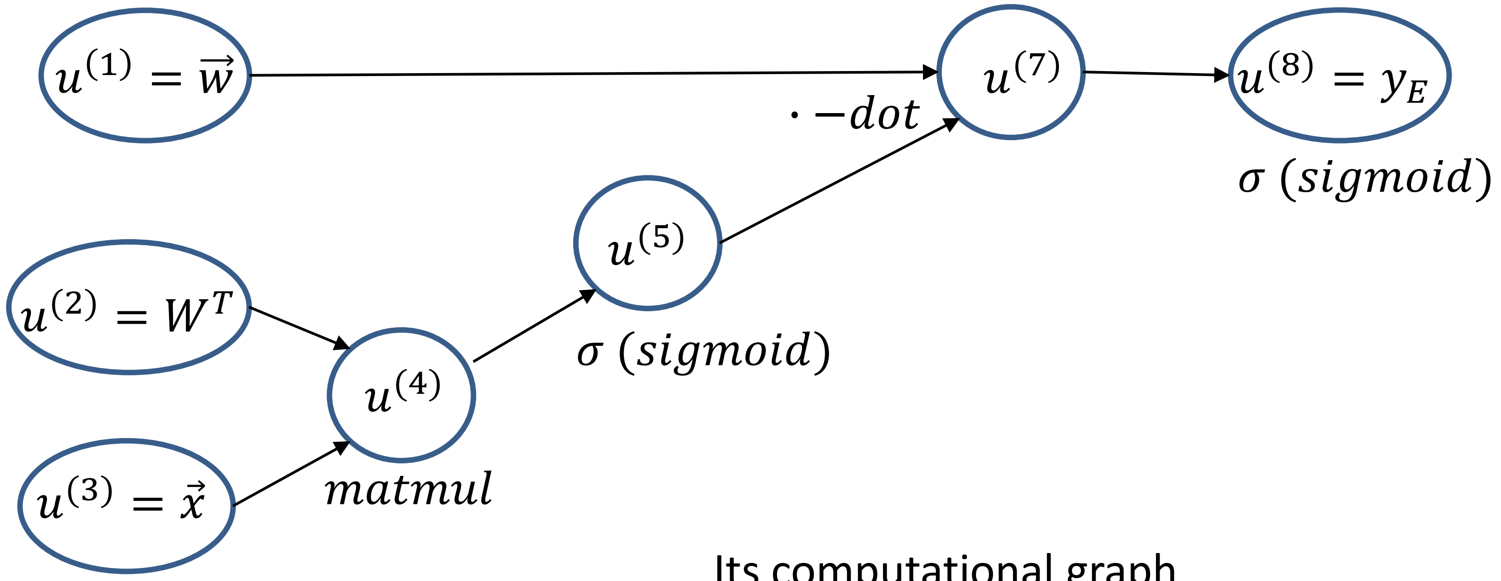


$$W = \begin{pmatrix} w_1 & w_2 \\ w_3 & w_4 \end{pmatrix}$$

$$\vec{w} = \begin{pmatrix} w_5 \\ w_6 \end{pmatrix}$$

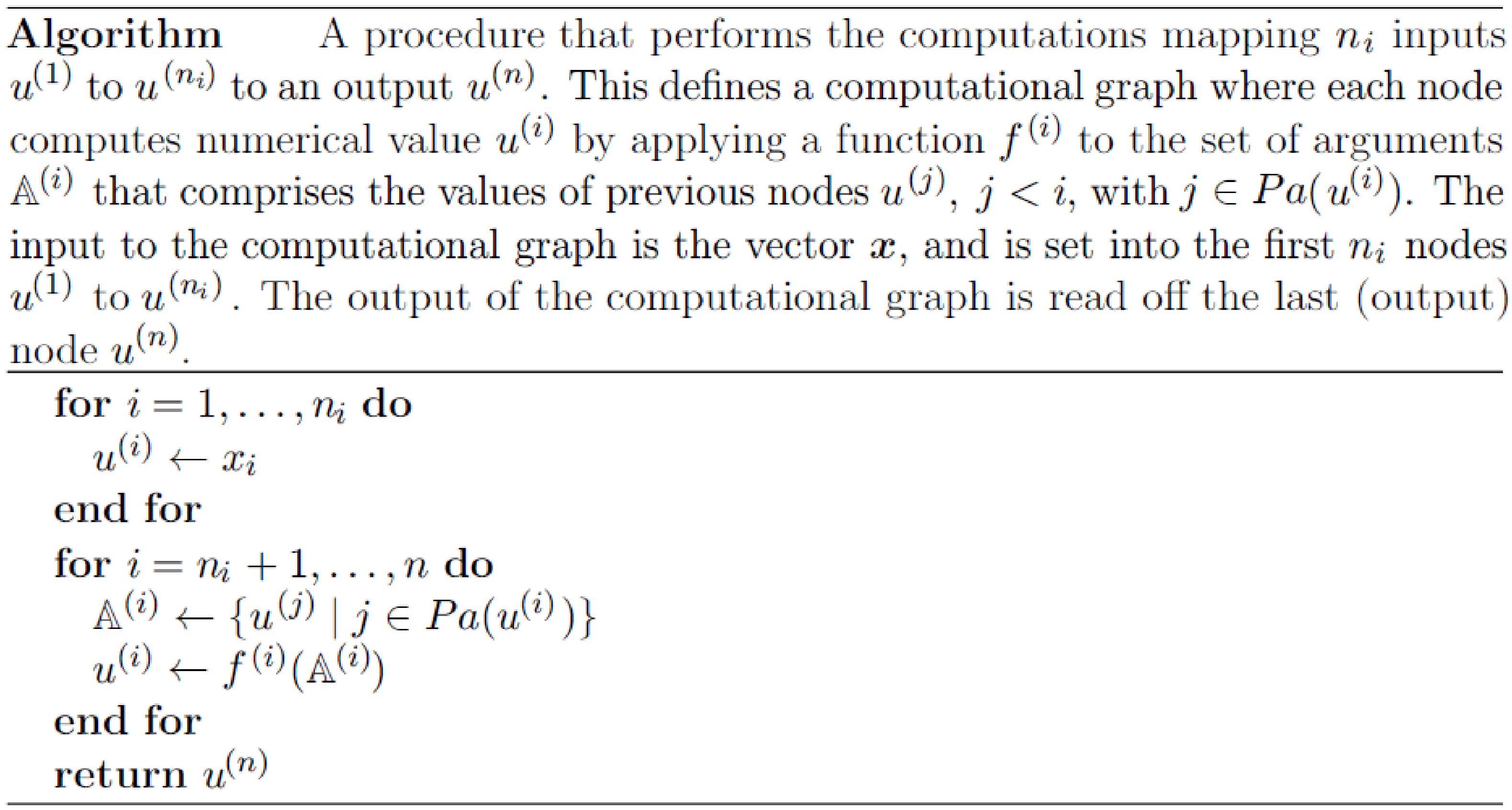
$$\vec{x} = \begin{pmatrix} x_A \\ x_B \end{pmatrix}$$

Neural net



Its computational graph

Forward Propagation Algorithm (step i.)



Algorithm A procedure that performs the computations mapping n_i inputs $u^{(1)}$ to $u^{(n_i)}$ to an output $u^{(n)}$. This defines a computational graph where each node computes numerical value $u^{(i)}$ by applying a function $f^{(i)}$ to the set of arguments $\mathbb{A}^{(i)}$ that comprises the values of previous nodes $u^{(j)}$, $j < i$, with $j \in Pa(u^{(i)})$. The input to the computational graph is the vector $\boldsymbol{x}$, and is set into the first n_i nodes $u^{(1)}$ to $u^{(n_i)}$. The output of the computational graph is read off the last (output) node $u^{(n)}$.

```
for $i = 1, \ldots, n_i$ do
    $u^{(i)} \leftarrow x_i$
end for
for $i = n_i + 1, \ldots, n$ do
    $\mathbb{A}^{(i)} \leftarrow \{u^{(j)} \mid j \in Pa(u^{(i)})\}$
    $u^{(i)} \leftarrow f^{(i)}(\mathbb{A}^{(i)})$
end for
return $u^{(n)}$
```

Auxiliary Graph for Backpropagation

- We construct a new graph $\mathcal{B}$ that is a reverse of computational graph G but is labeled exactly the same way
- This graph is needed to compute

$$\nabla_{\boldsymbol{U}} L_i = \frac{dL_{\mathrm{i}}}{du_n} \cdot \nabla_{\boldsymbol{U}} u_n$$

by chain rule, and then

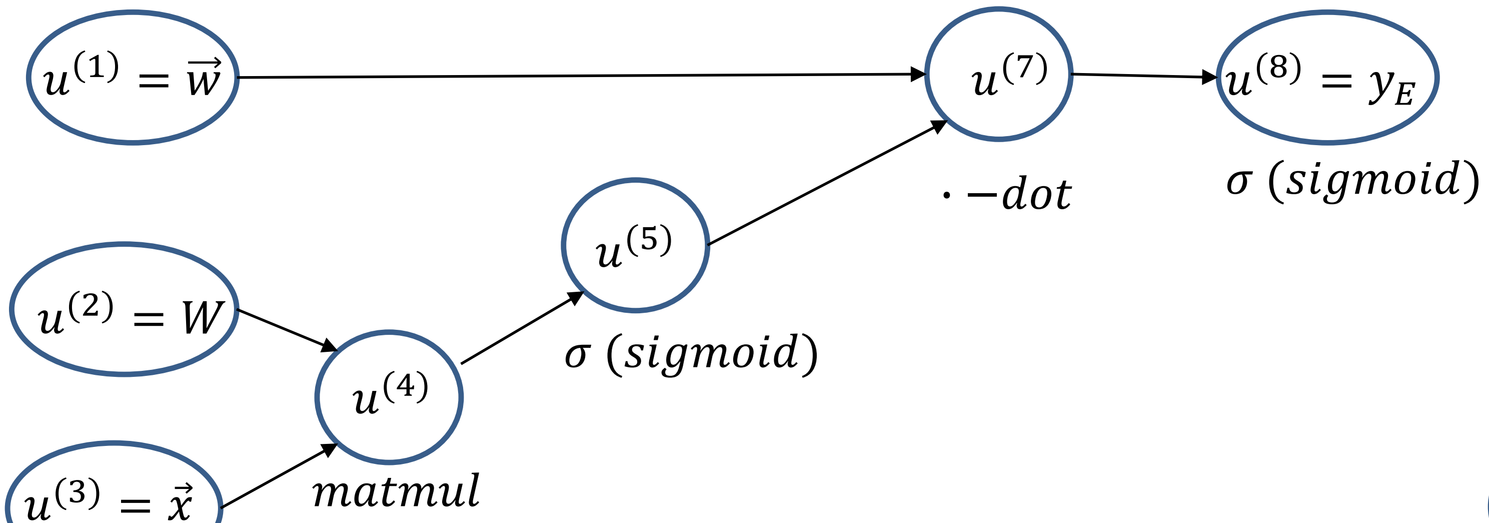
$$\nabla_{\boldsymbol{U}}\ u_i = \mathbb{J}_{\boldsymbol{U}}\left(\overrightarrow{\mathbb{A}(u_i)}\right)^T \nabla_{\overrightarrow{\mathbb{A}(u_i)}} u_i$$

where $\boldsymbol{U}$ is the tensor of all variables and $\mathbb{J}_{\boldsymbol{U}}$ is Jacobian of a vector function $\overrightarrow{\mathbb{A}(u_i)}$ We'll come back to it in more details

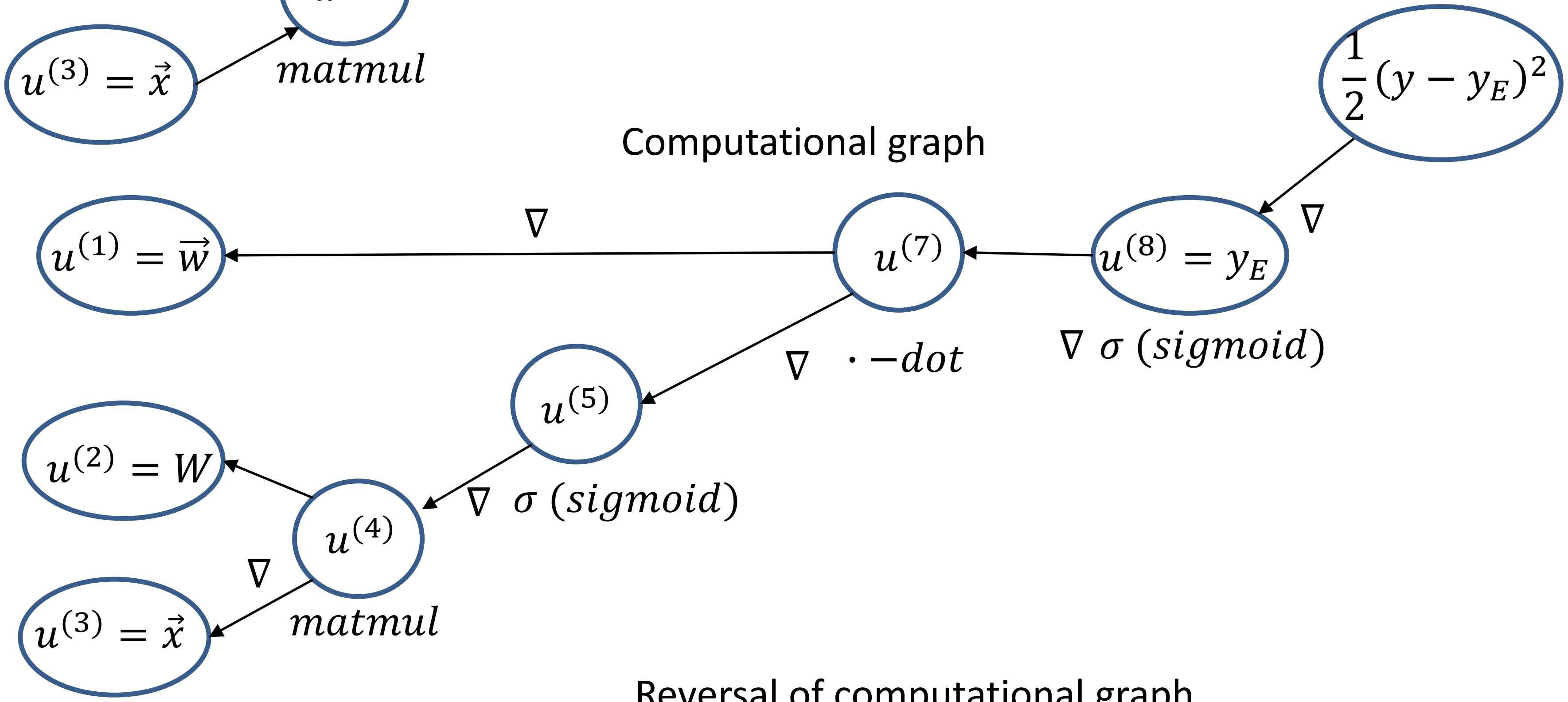
- Recall that $\mathbb{J}_{\vec{x}}\left(\vec{f}(\vec{x})\right) = \mathbb{J}_{\vec{x}}\left(\begin{bmatrix} f_1(\vec{x}) \\ \vdots \\ f_k(\vec{x}) \end{bmatrix}\right) = \begin{pmatrix} \frac{\partial f_1(\vec{x})}{\partial x_1} & \cdots & \frac{\partial f_k(\vec{x})}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1(\vec{x})}{\partial x_n} & \cdots & \frac{\partial f_k(\vec{x})}{\partial x_n} \end{pmatrix}$

- The edge from $u^{(i)}$ to $u^{(j)}$ in $\mathcal{B}$ is associated with the computation of $\frac{\partial u^{(i)}}{\partial u^{(j)}}$.

Computational Graph Reversal for Gradient

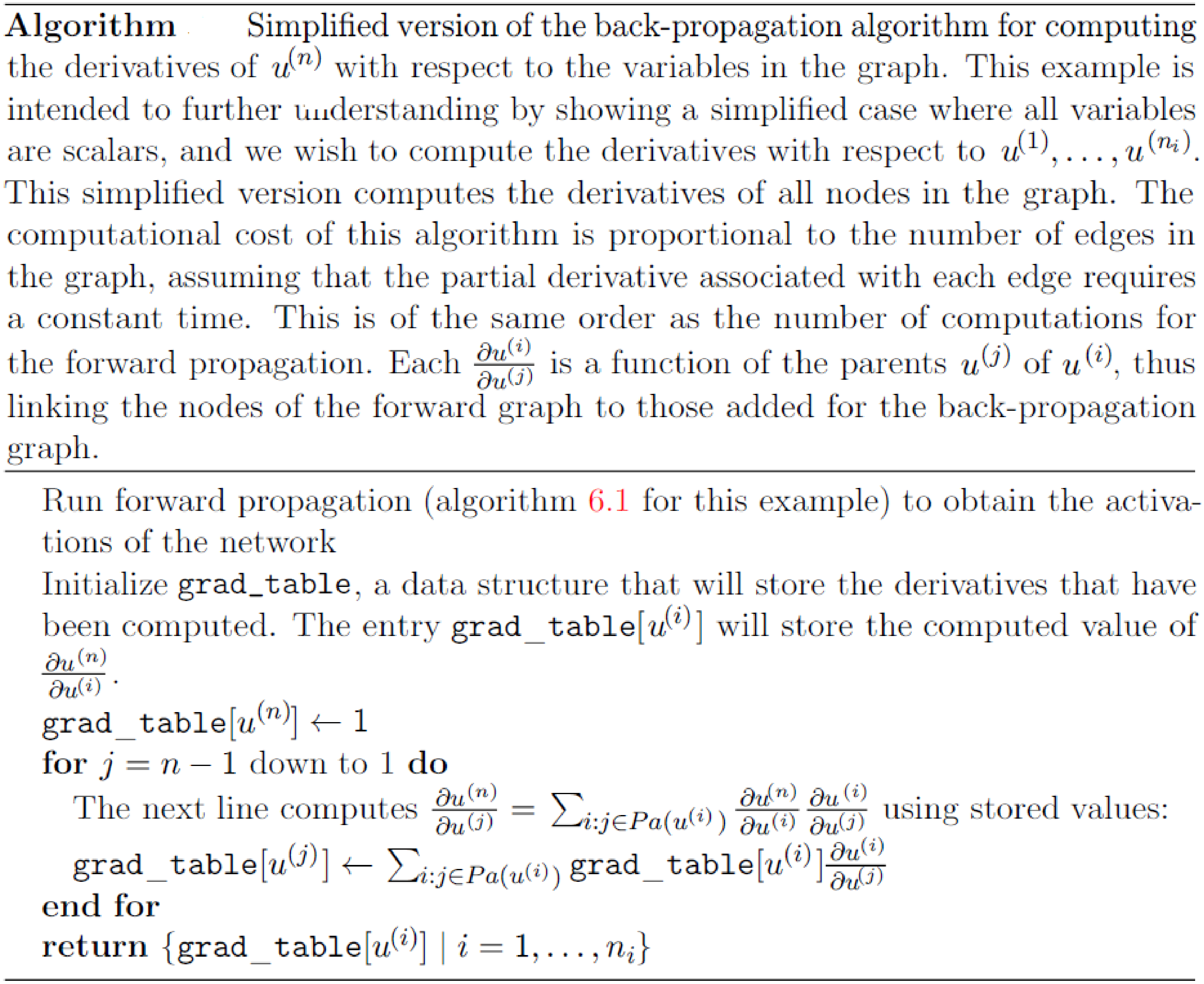


Computational graph



Reversal of computational graph

Backpropagation (step ii)



Algorithm Simplified version of the back-propagation algorithm for computing the derivatives of $u^{(n)}$ with respect to the variables in the graph. This example is intended to further understanding by showing a simplified case where all variables are scalars, and we wish to compute the derivatives with respect to $u^{(1)}, \ldots, u^{(n_i)}$. This simplified version computes the derivatives of all nodes in the graph. The computational cost of this algorithm is proportional to the number of edges in the graph, assuming that the partial derivative associated with each edge requires a constant time. This is of the same order as the number of computations for the forward propagation. Each $\frac{\partial u^{(i)}}{\partial u^{(j)}}$ is a function of the parents $u^{(j)}$ of $u^{(i)}$, thus linking the nodes of the forward graph to those added for the back-propagation graph.

```
Run forward propagation (algorithm 6.1 for this example) to obtain the activations of the network
Initialize grad_table, a data structure that will store the derivatives that have been computed. The entry grad_table[$u^{(i)}$] will store the computed value of $\frac{\partial u^{(n)}}{\partial u^{(i)}}$.
grad_table[$u^{(n)}$] $\leftarrow 1$
for $j = n - 1$ down to 1 do
  The next line computes $\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j \in Pa(u^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}}$ using stored values:
  grad_table[$u^{(j)}$] $\leftarrow \sum_{i:j \in Pa(u^{(i)})}$ grad_table[$u^{(i)}$]$\frac{\partial u^{(i)}}{\partial u^{(j)}}$
end for
return {grad_table[$u^{(i)}$] | $i = 1, \ldots, n_i$}
```

What’s left untouched

- Clearly this is divide and conquer algorithms
    - May instead include lookup
    - Clearly it is amenable to dynamic programming
- How do we do taking derivatives?
    - symbolic algebraic manipulation – term rewriting systems
    - Finite differences approximations
        - What’s magnitude?

Reading

- Chapter 1.3
- Chapters 3.2.1-3.2.3 (we'll comeback to these)
- CBC 6.5.1 - 6.5.8